CS 302 Homework 2 – due 3:30 p.m., Wednesday, April 28, 1999

Interpreting PCAT Programs

Working individually or in teams of two, write an interpreter for (a subset of) the PCAT language, as described in the *PCAT Language Reference Manual* (available from the course web page). *Exclude* anything to do with real numbers, and the `LOOP` and `EXIT` statements. The official abstract syntax for this subset of **PCAT** is described in `/u/cs302acc/ast.txt`. The Manual fails to indicate the range of the PCAT `INTEGER` type; it is intended to be a 32-bit signed value, i.e., in the range [-2147483648,2147483647].

The interpreter should be an executable that takes a single command line argument specifying the `.ast` file to be interpreted. It may assume that the `.ast` file has successfully passed the typechecker. The interpreter should execute the user program, reading any user input from `stdin` and writing any user output to `stdout`.

If the user program terminates normally, the interpreter should print the message "`Interpreter done`" to `stderr`. In addition, the interpreter should terminate with an appropriate error message to `stderr` if any of the following conditions occur: division by zero, failure to return a value from a function procedure, invalid user input to a `READ` statement, array subscript out of bounds, dereferencing a `NIL` record pointer. The interpreter may also terminate with the message "`Out of memory`" to `stderr` when appropriate. When given a non-terminating user program the interpreter should not terminate, or else give "`Out of memory`," or else dump core due to stack overflow. (Note that maximum stack size can be controlled using the `limit` shell command.) Under no other circumstances should the interpreter dump core on any typechecked program.

A "correct" interpreter executable is available in `/u/cs302acc/2/interp`.

Implementation

Package your interpreter as described above, to read the externalized AST format from a file specified on the command line. You can generate the type-checked `.ast` files however you like. One way is to use the front end (lexer, parser, typechecker) executable provided in `/u/cs302acc/bin/frontend`, which prints (type-checked) trees to stdout. Another is to use your own front-end code, using the print utilities in `/u/cs302acc/2/ast.c` to output the trees. In either case, use the parser provided in `/u/cs302acc/2/parse_ast.y` to do the work of parsing the external format into the internal ast structures defined in `/u/cs302acc/2/ast.h`. Support functions for environments and the store are available in `/u/cs302acc/2/state.[ch]`.

There is also a skeletal interpreter making use of these support functions in `/u/cs302acc/2/interp0.c`, which is capable of handling very simple user programs such as:

```
PROGRAM IS
VAR x : INTEGER := 2;
BEGIN
  WRITE ("2 + 2 = ", x + x);
END;
```

Feel free to build your interpreter by extending this skeletal version.

A number of short PCAT programs suitable for testing your interpreter are in directory `/u/cs302acc/pcat`. You are encouraged to submit additional test programs for use by the class in this and subsequent assignments; please mail your candidate programs to `apt@cs.pdx.edu`, and I'll arrange to put them in the `pcat` directory. The directory `/u/cs302acc/2/tst` is set up to use the regression testing approach described in the separate handout of that name.

Submitting the Program

Prepare a `makefile` that builds your interpreter and produces an executable called `interp`. Submit your program by `mailing` a `shar` "bundle" containing *all* the relevant files to `cs302acc@cs.pdx.edu`.